UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Proba scrisă a examenului de licenţă
Informatică Română

VARIANTA 2

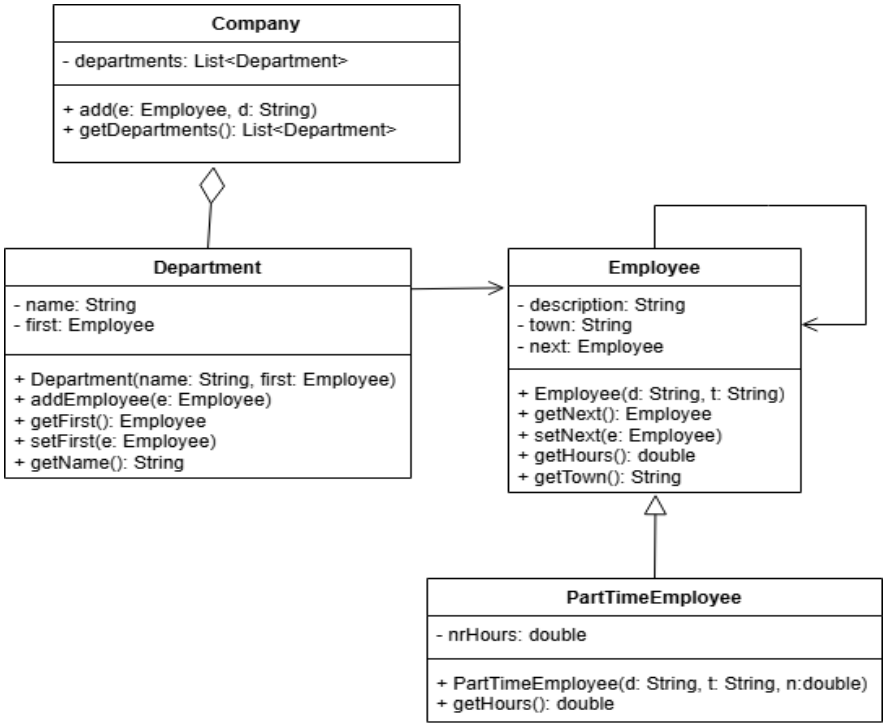
NOTĂ

- Toate subiectele sunt obligatorii. La toate subiectele se cer rezolvări cu soluții complete.
- Nota minimă ce asigură promovarea este 5.00.
- Timpul efectiv de lucru este de 3 ore.

SUBIECT Algoritmică şi programare

- **Se va indica limbajul de programare folosit.**
- **În implementările cerute nu este necesară dealocarea zonelor de memorie alocate dinamic.**
- **Lipsa unui stil de programare adecvat (denumiri sugestive pentru variabile, indentarea codului, comentarii dacă sunt necesare, lizibilitatea codului) duce la pierderea a 10% din punctajul aferent subiectului.**
- **Nu adăugați alte atribute, metode, în afara celor menționate în enunț, cu excepția constructorilor și a destructorilor, dacă e cazul. Nu modificați vizibilitatea atributelor specificate în enunț.**
- **Nu se vor folosi containere sortate, operații de sortare și căutare predefinite.**
- **Pentru tipurile de date puteți folosi biblioteci existente (Python, C++, Java, C#).**

1. (**2.5 puncte**) Definiți o funcție într-unul din limbajele de programare **Python, C++, Java** sau **C#**, care primește ca parametri o listă ordonată *lst* de șiruri de caractere în ordine alfabetică (reprezentând numele unor departamente din cadrul unei companii) și un șir de caractere *nume,* reprezentând numele unui departament. Presupunând (ca precondiție a funcției) că șirul de caractere *nume* nu apare în lista *lst*, funcția va adăuga în lista *lst* șirul de caractere *nume*. Adăugarea se va face astfel încât lista *lst* să rămână ordonată alfabetic după numele departamentelor. Se va folosi o implementare recursivă. Se pot implementa funcții auxiliare. Soluțiile iterative se punctează parțial.
2. (**1 punct**) Precizați complexitatea timp a funcției de la punctul **1.**, în cazurile *favorabil*, *mediu* și *defavorabil*. Justificați răspunsul.
3. (**5.5 puncte**) Se dă următoarea diagramă UML conținând clasele **Company** (Companie), **Department** (Departament, reprezentând o colecție de angajați), **Employee** (AngajatCuNormăÎntreagă), **PartTimeEmployee** (AngajatCuNormăRedusă).



- Atributele *description* și *town* din clasa **Employee** (descrierea postului și orașul de proveniență) și *name* din clasa **Department** (numele departamentului) trebuie să fie nevide, iar atributul *nrHours* din clasa **PartTimeEmployee** (număr lunar de ore ale angajatului) trebuie să fie număr strict pozitiv și mai mic decât 160. Constructorii trebuie să impună constrângerile.
- Clasa **Employee** are o metoda **getHours**() care returnează 160 și o metodă **getTown()** care returnează orașul de proveniență al angajatului. Metodele **getNext()**, respectiv **setNext(e: Employee)** returnează, respectiv setează, atributul *next*.
- Metoda **getHours**() din clasa **PartTimeEmployee** returnează numărul lunar de ore ale angajatului cu normă redusă (atributul *nrHours* din clasă).
- Clasa **Department** memorează o colecție de angajați (cu normă întreagă sau normă redusă) din departament. Metoda **addEmployee**(**e: Employee**) adaugă un angajat în departament, iar metoda **getName()** returnează numele departamentului. Metodele **getFirst()**, respectiv **setFirst(e: Employee)** returnează, respectiv setează, atributul *first*.
- Clasa **Company** memorează o listă de departmente. Metoda **add(e**: **Employee, d**: **String)** adaugă angajatul *e* în departamentul având numele *d* (în cazul în care nu există un departament cu numele *d***,** acesta se va crea). Metoda **getDepartments**() returnează lista departamentelor din companie.

Scrieți un program într-unul din limbajele de programare **C++, Java**, sau **C#**, cu următoarele cerințe:

a) Declarați toate clasele, atributele și metodele conform diagramei de mai sus. Implementați doar următoarele metode:
  a1) Metoda **setNext**(**e**: **Employee**) din clasa **Employee**.
  a2) Constructorii claselor **Department** și **PartTimeEmployee**.
  a3) Metodele **addEmployee(e: Employee)** și **setFirst (e: Employee)** din clasa **Department.**
  Pentru metoda **addEmployee(e: Employee)** se cere o implementare având o complexitate timp în caz defavorabil $\theta(1)$. Soluțiile care nu se încadrează în complexitatea cerută se punctează parțial.
  a4) Metoda **add(e: Employee, d: String)** din clasa **Company**.

b) Definiți o funcție care primește ca parametru un obiect *c* de tip **Company,** calculează numărul maxim de angajați (*m*) care provin dintr-un același oraș și apoi afișează toate orașele din care provin *m* angajați (un oraș se va afișa o singură dată). Presupunând că numele orașelor au cel mult 20 caractere, se cere o implementare având o complexitate timp în caz mediu $\theta(n)$ (*n* fiind numărul de angajați ai companiei). Soluțiile care nu se încadrează în complexitatea cerută se punctează parțial. **Nu este permisă folosirea unor funcții predefinite în biblioteci pentru determinarea elementelor distincte dintr-o secvență.**

c) Construiți un obiect de tip **Company** în care adăugați 4 angajați, din care 2 cu normă redusă (la alegere). Apelați apoi funcția de la punctul **b)**.

SUBIECT Baze de date **VARIANTA 2**

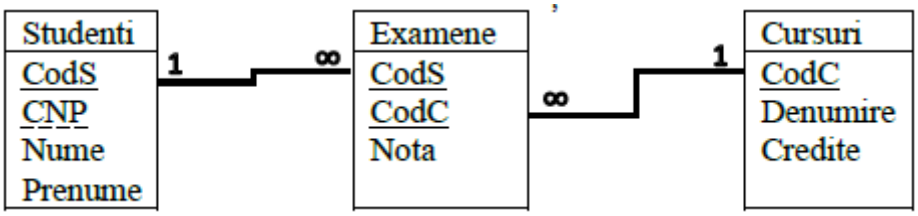
Problema 1. (4 puncte)

O companie software produce aplicații pentru diverse industrii folosind tehnologii adaptate fiecărei astfel de direcții. Pentru a gestiona alocarea angajaților din departamente pe proiectele realizate în firmă, compania utilizează o bază de date relațională:

- Un angajat are un cod, o dată a angajării, un nume (poate fi un atribut simplu, șir de caractere), o adresă de email (unică pentru fiecare angajat) și o valoare a salariului lunar; de asemenea, un angajat este alocat unui singur departament și unui singur proiect.
- Fiecare departament are un cod și o denumire; un departament poate avea de realizat mai multe proiecte în același timp.
- Un proiect are un cod, o denumire internă unică a proiectului, un buget și o tehnologie dominantă: de exemplu, proiectul cu denumirea internă P1T1 are ca tehnologie dominantă ".NET". El este realizat în cadrul unui singur departament.
- Compania dispune de mai multe locații unde angajații pot să își desfășoare activitatea: o locație are un cod, o adresă (poate fi un atribut simplu, șir de caractere) și număr de telefon (unic pentru fiecare locație); o locație are sau nu parcare, dispune sau nu de cafeterie.
- Într-o locație pot lucra angajați din mai multe departamente, dar pentru o locație a fost stabilit de la început ce departamente își desfășoară activitatea în acea locație. La o locație pot fi alocate mai multe departamente, iar unui departament îi pot fi alocate mai multe locații.

Realizați o schemă relațională BCNF pentru baza de date, evidențiind riguros cheile primare, cheile candidat și cheile externe. Realizați schema într-una din manierele indicate în exemplul de mai jos:
* exemplu pentru tabelele Studenti, Cursuri și Examene:

V1. Diagramă cu tabele, chei primare subliniate cu linie continuă, chei candidat subliniate cu linie întreruptă, legături trasate direct între cheile externe și cheile primare / candidat corespunzătoare (de exemplu, legătură trasată între coloana CodS din Examene și coloana CodS din Studenti).



V2.
Studenti[CodS, CNP, Nume, Prenume]
Cursuri[CodC, Denumire, Credite]
Examene[CodS, CodC, Nota]
Cheile primare sunt subliniate cu linie continuă, iar cheile candidat sunt subliniate cu linie întreruptă.
{CodS} este cheie externă în Examene și face referire la {CodS} din Studenti. {CodC} este cheie externă în Examene și face referire la {CodC} din Cursuri.

Problema 2. (5 puncte)

Se dau următoarele scheme relaționale:

Autori[CodAutor, Nume, Tara, AnulNasterii]

Carti[CodCarte, Titlu, Domeniu, NrPagini, *CodAutor*]

Edituri[CodEditura, Denumire]

CartiEdituri[*CodCarte*, *CodEditura*, Pret, AnAparitie]

Cheile primare sunt subliniate. Cheile externe sunt scrise cursiv și au aceeași denumire cu coloanele la care fac referire.

a. Scrieți o interogare SQL care returnează codul și numele fiecărui autor care a scris cel puțin 2 cărți într-un an și are cel puțin o carte publicată la editura "DE2".

b. Se dau instanțele următoare pentru relațiile Carti, Edituri și CartiEdituri:

Carti

CodCarte	Titlu	Domeniu	NrPagini	CodAutor
1	T1	D2	125	A1
2	T2	D1	175	A1
3	T3	D2	210	A3
4	T4	D3	98	A2
5	T5	D2	113	A4

Edituri

CodEditura	Denumire
E1	DE1
E2	DE2
E3	DE3

CartiEdituri

CodCarte	CodEditura	Pret	AnAparitie
1	E2	34	2021
2	E1	28	2022
3	E3	55	2021
4	E2	21	2024
5	E1	42	2023

b1. Precizați rezultatul evaluării interogării de mai jos pe instanțele date. Menționați strict valorile tuplului / tuplurilor și denumirile coloanelor din rezultat fără a prezenta toți pașii evaluării interogării.

```
SELECT t1.CodAutor, t1.Titlu, t1.CodCarte
FROM Carti t1
WHERE
EXISTS
        (SELECT CodCarte FROM CartiEdituri t2
        INNER JOIN
        (SELECT ce.CodEditura FROM CartiEdituri ce
                GROUP BY ce.CodEditura
                HAVING COUNT(*)>1 AND SUM(ce.Pret)<60
        UNION ALL
        SELECT e.CodEditura FROM Edituri e WHERE e.Denumire='DE2'
        )t3 ON t3.CodEditura=t2.CodEditura
                WHERE t2.CodCarte=t1.CodCarte
        )
```

b2. Explicați dacă următoarele dependențe funcționale sunt satisfăcute sau nu de datele din instanța Carti:

- {NrPagini} → {CodCarte}
- {CodAutor} → {Domeniu}

VARIANTA 2

SUBIECT Sisteme de operare

Problema 1 (5 puncte). Răspundeţi la următoarele întrebări despre execuţia programului `./a` de mai jos, apelat iniţial fără argumente, considerând că toate include-urile necesare sunt prezente şi că apelul sistem `fork` şi cel puţin unul dintre apelurile sistem `exec` se execută cu succes.

```
 1  int main(int argc, char** argv) {
 2      if(argc > 1) {
 3          printf("%s\n", argv[1]);
 4      }
 5      fork();
 6      execlp("a", "a", "y", NULL);
 7      execl("./a", "./a", "z", NULL);
 8      printf("%d\n", wait(NULL));
 9      return 0;
10  }
```

a) Explicaţi în detaliu funcţionarea liniei 6.

b) De câte ori se va afişa în consolă valoarea `argv[1]`? Justificaţi răspunsul.

c) Ce se va afişa în consolă dacă variabila de mediu `PATH` nu include directorul curent?

d) Câte procese vor executa apelul sistem `wait`?

e) Ce se va afişa în consolă dacă variabila de mediu `PATH` include directorul curent şi se elimină linia 5?

Problema 2 (4 puncte). Răspundeţi la următoarele întrebări despre execuţia scriptului Shell UNIX de mai jos, considerând că fişierul `a.txt` există şi conţine pe fiecare linie secvenţe de litere, separate prin spaţii.

```
 1  #!/bin/bash
 2
 3  while read L; do
 4    R=""
 5    for W in $L; do
 6      X=""
 7      C=`echo $W | sed -E "s/(.)/\1 /g"`
 8      for A in $C; do
 9        X="$A$X"
10      done
11      R="$X $R"
12    done
13    echo $R
14  done < a.txt
```

a) Ce valori vor lua variabilele L şi W?

b) Explicaţi în detaliu funcţionarea liniei 7.

c) Ce va afişa în consolă rularea scriptului dacă fişierul `a.txt` are conţinutul de mai jos?

```
abc def
pqr xyz
```

d) Ce va afişa în consolă rularea scriptului pentru fişierul `a.txt` definit la punctul c), dacă se elimină din linia 7 spaţiul de după cifra `1`?

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1.** **– 2.5p**

- signatura – **0.2p**
- implementare recursivă – **2.3p** din care
  - adăugare lista vidă – 0.5
  - adăugare la începutul/sfârșitul listei (pentru caz favorabil) – 0.5
  - apelul recursiv – 1.3

* implementare iterativă corectă – **0.7**p

Cerința **2.** **– 1p**

- caz favorabil **0.2p** din care
  - complexitate $\theta(1)$ – 0.1
  - justificare – 0.1
- caz mediu **0.4p** din care
  - complexitate $\theta(n)$ – 0.2 (n – dimensiunea listei *lst*)
  - justificare – 0.2
- caz defavorabil **0.4p** din care
  - complexitate $\theta(n)$ – 0.2 (n – dimensiunea listei *lst*)
  - justificare – 0.2

Cerința **3.a)** **– 2.6p**

Definirea clasei Employee – **0.6p** din care
- atribute – 0.1
- contructor – 0.1
- metoda **setNext (a1)** – 0.2
- metode **getNext**, **getHours**, **getTown** – 0.2

Definirea clasei PartTimeEmployee– **0.55p** din care
- relația de moștenire – 0.2
- atribut – 0.05
- **constructor (a2)** – 0.2
- metoda **getHours** – 0.1

Definirea clasei Department– **1p** din care
- atribute – 0.1
- **constructor (a2)**– 0.1
- metoda **addEmployee (a3)** – 0.5
  - * punctajul se acordă pentru implementare având complexitate $\theta(1)$
  - * pentru implementari având complexitate mai mare decat $\theta(1)$ se acorda 0.25p
- metoda **setFirst (a3) –** 0.2
- metode **getFirst**, **getName –** 0.1

Definirea clasei Company – **0.45p** din care
- atribut, metoda **getDepartments** – 0.1p
- metoda **add (a4)** – 0.35p

Funcția **3.b)** – **2.6p**

- signatura – **0.1p**
- implementare având complexitatea timp în caz mediu $\theta(n)$ – **2.5p**
  - * implementare având complexitatea timp în caz mediu $\theta(n\log_2 n)$ – **1.5p**
  - * implementare având complexitatea timp în caz mediu $\theta(n^2)$ – **0.75p**

Funcția principală **3.c)** – **0.3p**

- construire obiect de tip Company – 0.1p
- adăugare angajați - 0.1p
- apel funcție b) pentru expresie– 0.1p

BAREM INFORMATICĂ
VARIANTA 2

Subiect Baze de date

Oficiu – **1p**

Problema 1. Punctaj - 4p

- relații cu atribute corecte, chei primare, chei candidat: **3p**
- legături modelate corect (chei externe): **1p**

Problema 2. Punctaj - 5p

- **a** - rezolvarea completă a interogării: **2.5p**

- **b1** - rezultat evaluare interogare:

CodAutor	Titlu	CodCarte
A1	T1	1
A2	T4	4

  - coloane – **0.5p**
  - valori tuplu – **1p**

- **b2** - {NrPagini}→{CodCarte} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodAutor}→{Domeniu} nu este satisfăcută – **0.25p**; **0.25p** explicație

Notă: La specializările Informatică engleză și Informatică maghiară se iau în considerare versiunile traduse în limbile corespunzătoare.

VARIANTA 2

SUBIECT Sisteme de operare

Barem:

1p – oficiu

Problema 1 (5p)

1p – a) Execută programul `a` cu argumentul `y`, dacă locaţia programului `a` se găseşte in `PATH`

1p – b) Se va afişa de o infinitate de ori, pentru că programul se reexecută la linia 6 sau 7.

1p – c) Se vor afişa o infinitate de caractere `z`

1p – d) Nicunul pentru că unul dintre cele două apeluri sistem `execl` şi `execlp` se vor executa cu succes şi vor suprascrie codul procesului înainte să se ajungă la linia 8.

1p – e) Se vor afişa o infinitate de caractere `y`

Problema 2 (4p)

1p – a) L va lua ca valori fiecare linie din fişierul a.txt, variabila W va lua ca valori fiecare secvenţe de litere din linia stocată în L

1p – b) După fiecare literă din variabila W se adaugă un spaţiu, rezultatul stocându-se in variabila C

1p – c) Se va afişa

```
fed cba
zyx rqp
```

1p – d) Se va afişa

```
def abc
xyz pqr
```